

Online Knapsack Problem and Budgeted Truthful Bipartite Matching

Rahul Vaze

School of Technology and Computer Science

Tata Institute of Fundamental Research

Mumbai, India

vaze@tcs.tifr.res.in

Abstract—Two related online problems: knapsack and truthful bipartite matching are considered. For these two problems, the common theme is how to ‘match’ an arriving left vertex in an online fashion with any of the available right vertices, if at all, so as to maximize the sum of the value of the matched edges, subject to satisfying a sum-weight constraint on the matched left vertices. Assuming that the left vertices arrive in an uniformly random order (secretary model), two almost similar algorithms are proposed for the two problems, that are $2e$ competitive and 24 competitive, respectively. The proposed online bipartite matching algorithm is also shown to be truthful: there is no incentive for any left vertex to misreport its bid/weight. Direct applications of these problems include job allocation with load balancing, generalized adwords, crowdsourcing auctions, and matching wireless users to cooperative relays in device-to-device communication enabled cellular network.

I. INTRODUCTION

In this paper, we consider two basic online combinatorial problems : knapsack and truthful bipartite matching, that have wide applications in practice. We first consider the *online* knapsack problem, where each item, that has two attributes : value and weight, appears sequentially, and has to be accepted/rejected irrevocably using only causal information, to maximize the total value of the selected items subject to the sum of their weights being less than the specified capacity. The knapsack problem is a classical combinatorial problem, whose online version has also received considerable attention in the literature [1]–[3], as it captures many of the modern resource allocation problems such as generalized adwords, job allocation in cloud computing, load balancing, cognitive radio, admission control and many others [4]–[8].

The second and related problem to the online knapsack problem is the truthful budgeted bipartite matching problem over a graph $G(L \cup R, E)$, where the right vertex set R is known ahead of time, while left vertices of L arrive sequentially. On the arrival of a left vertex ℓ , utilities of all edges incident on it as well as its bid $c(\ell)$ are revealed. Any left vertex can be matched or accepted only if the payment made to it is larger than $c(\ell)$. With a total payment budget constraint of C , the problem is to decide which unmatched vertex of R to match with ℓ , if at all, immediately and irrevocably, so as to maximize the sum of the utility of all the matched/accepted edges. We assume that left vertices are strategic players, which could potentially manipulate the reporting of their true bid, and hence seek a truthful algorithm, i.e., no incoming vertex has any incentive to misreport its bid to maximize its profit.

The two problems are closely related, since knapsack problem can be modelled as a bipartite matching problem, where all edges incident on a left vertex have same utilities (value of the item) and the capacity constraint on sum-weight is equivalent to the payment budget constraint. Only the truthful aspect is different.

Important applications of the truthful budgeted bipartite matching problem are in crowdsourcing [9], [10] and device-to-device (D2D) cellular wireless communication. The crowdsourcing motivation is exemplified by modern cloud platforms such as Amazon’s Mechanical Turk (MTRK), ClickWorker (CLKWRKR), CrowdFlower (CRDFLWR) that has been well discussed in literature [9]–[12]. In a D2D network, the basic idea is for *idle* nodes to help relay other nodes’ data to/from the basestation or amongst themselves [13], [14]. Since relaying costs resources, each node demands a payment for its help, and the problem is to find an association/matching rule as to who should help whom [15] and also the payment to be made for helpers, subject to a total budget constraint on payment. To extract largest payment, each node can behave strategically, and hence there is a need for making this association/matching truthful.

To keep both the problems non-degenerate, similar to other prior related works on online algorithms [3], [16], we consider a secretarial input model, where the order of arrival of items/left vertices is uniformly random, but their utilities and bids are allowed to be arbitrary. Under this model, we first consider an offline algorithm proposed in [17] that is useful for both problems, and then use the sample and price idea to make the algorithms *online*. We also make a large market assumption, i.e., the utility of any one edge is small compared to the sum-utility of the optimal matching, that is commonly observed in practice for most problems of interest, and is widely used in auction literature [18]–[20].

To quantify the performance of any online algorithm, we use the well established metric of *competitive ratio*, that measures the ratio of the profit of the online algorithm and the optimal offline algorithm (that has access to non-causal information).

We briefly discuss the prior work on both these problems. The online knapsack problem has been studied widely [1]–[3], with the best known competitive ratio of $10e$ reported in [3] for a randomized algorithm under the secretarial input. The truthful budgeted bipartite matching problem is a special case of a reverse auction [21], where users submit bids for accomplishing a set of tasks and if selected, expect a payment

at least as much as their reported bids. The offline version of the truthful matching problem, where the full graph is revealed ahead of time, has been considered in [12], where a 3-approximate algorithm has been derived that is one-sided truthful. When the goal is to maximize the number of matched edges, [22] provides a 320-competitive online truthful algorithm assuming the secretarial input model. Under large market assumption, the best known bound for the considered online problem is a 24β -competitive algorithm [17], where β is the ratio of the largest to the smallest utility of any edge. Under some additional restrictions such as utilities of all edges incident on a right vertex are identical, a constant-competitive algorithm has been derived in [23]. Our contributions:

- Assuming a large market assumption and secretarial input, we propose a simple algorithm for the online knapsack problem, that is shown to be $2e$ competitive. Compared to prior work [3], enforcing the large market assumption, which is mostly satisfied in practice especially in networking problems, we are able to significantly improve the competitive ratio from $10e$ to $2e$. Moreover, our algorithm is also deterministic.
- The second main contribution of this paper is a 24 -competitive online bipartite matching algorithm that is truthful and satisfies the payment budget constraint. The previous best known result is a 24β -competitive algorithm [17] (β is the ratio of the largest to the smallest utility of any edge). Since our algorithm has constant competitiveness, it is scalable and appealing for applications in large networks.

II. ONLINE KNAPSACK PROBLEM

Let the value and weight of item $i \in \mathcal{I}, |\mathcal{I}| = n$, be $v(i)$ and $w(i)$, respectively, and the corresponding weight to value ratio (called the buck per bang in the paper) be $b(i) = \frac{w(i)}{v(i)}$. The weights and values (and buck per bang) are arbitrary and allowed to be selected by an adversary. The knapsack problem is to select the set of items that maximizes the sum of their values, subject to a constraint C on the sum of the weight of the items in the selected set. Thus, without loss of generality, let $w(i) \leq C, \forall i$.

We consider the online knapsack problem, and to keep it non-degenerate in terms of competitive ratio, we assume that the order of arrival of items is uniformly random (secretary-model), i.e., each permutation over n arriving items is equally likely. Let π be a uniformly random permutation over $[1 : n]$. Then the k^{th} item that arrives has value $v(\pi^{-1}(k))$, weight $w(\pi^{-1}(k))$, and buck per bang $b(\pi^{-1}(k))$. Under this model, we also assume that given two items arriving at locations $\pi(i)$ and $\pi(j)$, if $b(i) > b(j)$, then $P(w(i) > w(j)) = \frac{1}{2}$ which is reasonable for most applications.

For a set S , we let $v(S) = \sum_{s \in S} v(s)$. For any online algorithm A (where on arrival of item i , it has to be either accepted/rejected instantaneously and irrevocably), the competitive ratio for solving the knapsack problem is defined as

$$\mu_A = \min_{\mathcal{I}} \frac{\mathbb{E}_{\pi} \left\{ \sum_{s \in S_A} v(s) \right\}}{v(\text{OPT}(C))},$$

where $\text{OPT}(C)$ is the optimal offline set of selected items and S_A is the set of items selected by A , with sum weight constraint C . The online knapsack problem is to find the best algorithm A that maximizes the competitive ratio μ_A . A is said to be $\alpha > 1$ competitive if $\mu_A = 1/\alpha$.

We map the knapsack problem to a matching problem,¹ where we define a bipartite graph $G = (L \cup R, E)$ whose each left vertex $\ell \in L$ corresponds to item $\ell \in \mathcal{I}$ ($|L| = |\mathcal{I}|$), and the number of right vertices $|R| = |\mathcal{I}|$, and edge set $E = \{e = (\ell, r) : v(e) = v(\ell), \forall r \in R\}$. Thus, each edge incident on left vertex ℓ has the same value. Finding the max-weight matching M in G in an online manner, such that $\sum_{e=(\ell, r) \in M} w(\ell) \leq C$ is equivalent to solving the online knapsack problem, where on arrival of each left vertex it has to be matched or permanently left unmatched, instantaneously and irrevocably. From hereon, we entirely focus on finding an efficient bipartite online matching subject to capacity constraint C .

Assumption 1. Let $v_{max} = \max_{e \in E} v(e)$, and $v(\text{OPT}(C))$ be the optimal value of the matching under the capacity constraint. We assume the typical large market assumption [12], i.e., $\frac{v_{max}}{v(\text{OPT}(C))} = o(1)$, thus, no single user can influence the outcome significantly.

Similar to buck per bang of left vertex, we define for each edge $e = (\ell, r)$ a buck per bang $b(e) = \frac{w(e)}{v(e)}$ that represents the weight/cost per unit utility. For any γ , let $G(\gamma)$ be the graph obtained by removing all edges $e \in E(G)$ with buck per bang $b(e) > \gamma$. Then the proposed online max-weight algorithm ON for solving the online knapsack problem is as given by Algorithm 1. The idea behind ON is as follows:

- Do not match any of the first t left vertices (called the offline phase), and only use them to run the offline THRESHOLD algorithm [17] and find the threshold γ_t and the matching M_t with capacity C .
- For any right vertex r , such that $e = (*, r) \in M_t$, set its price(r) and cost(r) to be the buck per bang and the weight of the left vertex matched to r in M_t , respectively.
- In the decision phase, starting with the arrival of $t + 1^{st}$ left vertex, do not consider it for selection if its buck per bang $b(e)$ larger than γ_t . Otherwise, match the newly arrived left vertex ℓ to the available/unmatched right vertex r with the smallest price that is larger than the buck per bang $b(\ell)$ of ℓ and has weight less than the cost of r . Thus the number of selected/matched left vertices is at most the number of left vertices matched by the THRESHOLD algorithm in the offline phase.

Before proving results on ON, we first consider the subroutine (THRESHOLD algorithm [17]) that is used to generate an offline matching with the first t left vertices, where the GREEDY subroutine is the usual greedy matching algorithm for a bipartite graph. Essentially, the THRESHOLD algorithm tries to find the largest threshold γ_C such that the sum-weight of the edges that are part of the greedy matching on the

¹The degree of any left or right vertex can be at most 1.

Algorithm 1 ON Algorithm

```
1: Input:  $L$  set of left vertices/users that arrive sequentially
   in order  $\pi$ ,  $R$  set of right vertices, Capacity  $C$ 
2: %Offline Phase
3:  $L_t$  = first  $t$  left vertices of  $L$ 
4: Run THRESHOLD on  $G_t = (L_t \cup R, E_t)$  to obtain  $\gamma_t \triangleq$ 
    $\gamma_C(G_t)$  and matching  $M_t$ 
5: for each right vertex  $r \in R$  do
6:   if  $e = (\ell, r) \in M_t$  then
7:     Set  $\text{price}(r) := b(\ell)$ ,  $\text{cost}(r) := w(\ell)$ 
8:   else
9:      $\text{price}(r) := 0$ ,  $\text{cost}(r) := 0$ 
10:  end if
11: end for
12: %Decision Phase
13:  $M_{\text{ON}} = \emptyset$ 
14:  $R' = \{r \in R : \text{price}(r) > 0\}$ .
15: for every new left vertex  $\ell \in L \setminus L_t$  do,
16:   if  $b(\ell) = \frac{w(\ell)}{v(\ell)} > \gamma_t$  then
17:     %Pruning: Let  $\ell$  be permanently unmatched
18:     Break
19:   else
20:     Let  $e^* = (\ell, r)$  be the edge with the smallest
      $\text{price}(r), r \in R'$  such that  $b(\ell) < \text{price}(r)$  and  $w(\ell) <$ 
      $\text{cost}(r)$ 
21:     if  $M_{\text{ON}} \cup \{e^*\}$  is a matching then
22:        $M_{\text{ON}} = M_{\text{ON}} \cup \{e^*\}$ 
23:     else
24:       Let  $\ell$  be permanently unmatched,
25:     end if
26:   end if
27: end for
```

edges with buck per bang less than the threshold, satisfies the capacity constraint.

Algorithm 2 THRESHOLD

```
1: Input: Graph  $G$ , Capacity  $C$ 
2: Output: Matching  $M$ , Threshold  $\gamma_C$ 
3:  $\mathcal{A}(G) = \{\gamma : \sum_{e \in M} \gamma v(e) \leq C, M = \text{GREEDY}(G(\gamma))\}$ 
4:  $\gamma_C = \max\{\gamma : \gamma \in \mathcal{A}(G)\}$ 
5: Accept all users in  $M = \text{GREEDY}(G(\gamma_C))$ 
```

Remark 1. The matching M output by THRESHOLD algorithm for graph G is a Greedy matching for graph $G(\gamma_C)$. Moreover, since all matched left vertices have $b(e) \leq \gamma$, and from the definition of THRESHOLD algorithm, $\gamma_C \sum_{e \in M} v(e) \leq C$, we have $\sum_{\ell: e=(\ell, r) \in M} w(\ell) \leq C$, i.e., M satisfies the capacity constraint.

We next list some important properties of THRESHOLD algorithm [17], whose proofs are presented in the Appendices for completeness sake.

Lemma 1. [17] Let $M(\text{off})$ be the matching output by THRESHOLD algorithm with input graph G under capacity constraint C . Then under Assumption 1, $v(M(\text{off})) \geq \frac{\text{OPT}(C)}{3+o(1)}$.

Lemma 1 is valid for all graphs, but if we restrict to a special class of graphs considered in this section, where values of all edges incident on any left vertex are identical and the number of right and left vertices are equal, we can get a better bound as a corollary to Lemma 1 as follows.

Corollary 1. Let $M(\text{off})$ be the matching output by THRESHOLD algorithm with input graph G (where edge set $E = \{e = (\ell, r) : v(e) = v(\ell)\}$) under capacity constraint C . Then under Assumption 1, $v(M(\text{off})) \geq \frac{\text{OPT}(C)}{1+o(1)}$.

Remark 2. Assumption 1 is critical in the sense that if it is violated, then the approximation ratio of the THRESHOLD algorithm can be arbitrarily bad which can be showed as follows. Consider the case when there are only two items, $v(1) = 1, w(1) = 1$, and $v(2) = C - 1, w(2) = C$, with capacity C . The optimal solution is to just choose item 2 (assumption 1 is not satisfied since $v(\{2\})/v(\text{OPT}(C)) = 1$), while the THRESHOLD algorithm will choose item 1 and the approximation ratio will be $1/C$.

Before analyzing the ON algorithm, we first consider the offline case, when THRESHOLD is run over the full graph $G(L \cup R, E)$ and output threshold is γ and matching is $M(\text{off})$. Recall that the edge weights of all edges incident on any left vertex are identical and the number of left and right vertices are equal. Hence the greedy matching $M(\text{off})$ output by the THRESHOLD 'offline' algorithm (when run on the full graph $G(L \cup R, E)$) contains all the left vertices that have buck per bang less than or equal to the threshold γ . Let the set of left vertices selected by the THRESHOLD algorithm be L^* , i.e., set of left vertices with buck per bang less than γ . From Corollary 1, we know that the utility of set L^* is almost optimal.

In the online case, we now aim to select as many left vertices of L^* , though without knowing γ exactly, since THRESHOLD cannot be run on the full graph G . Alternatively, we are trying to select as many left vertices that have buck per bang less than γ . This is reminiscent of the k -secretary problem, where the objective is to select the k secretaries with the largest utilities in an online fashion.

Apart from the major challenge of finding γ , another minor problem is that we do not know how many secretaries we want to pick ahead of time. We overcome both these challenges via algorithm ON, where we first estimate a $\gamma_t \geq \gamma$ by running THRESHOLD on a subgraph $G_t \subseteq G$ (graph consisting of the first t left vertices of G), and then select as many left vertices that are matched/selected by running THRESHOLD on graph G_t . We show that algorithm ON selects any left vertex that is part of L^* with probability at least $1/2e$.

We next state a critical lemma for analyzing the performance of the ON algorithm that shows that the γ_t computed in the offline phase of ON is always larger than γ (Lemma 4), and hence all vertices of G that are part of $M(\text{off})$ are not pruned in Step 17 of the ON algorithm.

Lemma 2. [17] Let $G = (L \cup R, E)$ and $F \subseteq G$, such that $F = (L \setminus L' \cup R, E')$, and the edge set E' is such that all edges incident on left vertices in set L' are removed simultaneously,

while all edges incident on $L \setminus L'$ are retained as it is. Then

$$v(\text{GREEDY}(G)) \geq v(\text{GREEDY}(F)).$$

Moreover

$$v(\text{GREEDY}(G(\gamma_1))) \geq v(\text{GREEDY}(G(\gamma_2))) \text{ for } \gamma_1 \geq \gamma_2,$$

and

$$v(\text{GREEDY}(G(\gamma))) \geq v(\text{GREEDY}(F(\gamma))).$$

For arbitrary subgraph $F \subseteq G$ (where any arbitrary edges are removed from G), $v(\text{GREEDY}(G))$ may or may not be larger than $v(\text{GREEDY}(F))$. The importance of Lemma 2 is in showing that THRESHOLD is solvable in polynomial time and the threshold γ_C is monotonic for classes of graphs considered in this paper. In particular, for the bipartite graphs considered in this paper, each left vertex has a fixed weight/cost and the buck-per-bang of edge $e = (\ell, r)$ is $b(e) = \frac{w(\ell)}{v(e)}$. Thus, if any edge $e = (\ell, r)$ has $b(e) > \gamma$, then all edges $e' = (\ell, *)$ for which their value $v(e') < v(e)$ that are incident on the left vertex ℓ also have $b(e) > \gamma$ and are not part of graph $G(\gamma)$. We prove the two claims as follows.

Lemma 3. [17] THRESHOLD is solvable in polynomial time.

Algorithm THRESHOLD involves finding a maximum in Step 4. In the proof, it is shown that bisection can be used to solve this maximization. We would like to note that if $v(\text{GREEDY}(G(\gamma))) \not\geq v(\text{GREEDY}(F(\gamma)))$, then finding this maximum is non-trivial.

The following Lemma shows that if THRESHOLD algorithm is run on a (special) subgraph of G , then the output threshold γ_C increases, which we critically need to show that all left vertices that are part of L^* are eligible for matching in the ON algorithm.

Lemma 4. [17] Let $G = (L \cup R, E)$ and $F = (L \setminus L' \cup R, E')$, where the edge set E' is such that all edges incident on left vertices in set L' are removed simultaneously, while all edges incident on $L \setminus L'$ are retained as it is. Then $\gamma_C(F) \geq \gamma_C(G)$.

Finally, we are ready to state the first main result of the paper on the expected utility of the online matching M_{ON} , output by the ON algorithm.

Theorem 1. $\mathbb{E}\{v(M_{\text{ON}})\} \geq \frac{v(\text{OPT}(C))}{2e(1+o(1))}$.

Proof: Consider the full graph $G = (L \cup R, E)$ (offline) and its subset $G_t = (L_t \cup R, E_t)$ (offline for ON), and let γ and γ_t be the output threshold when THRESHOLD is run over G and G_t , both with capacity C , respectively. From Lemma 4, it follows that $\gamma_t \geq \gamma$, hence all the left vertices L^* matched by the THRESHOLD algorithm with the full graph G that arrive in the decision phase are not pruned in Step 17 with the ON algorithm.

In the decision phase of the ON algorithm, disregard the condition that $w(\ell) < \text{cost}(r)$ for selecting a left vertex for now. Then the left vertex $\ell \in L^*$ that appears in the decision phase at the i^{th} position, $i > t$, is selected as long as it is selected by the VIRTUAL Algorithm [3]. This assertion follows since with the VIRTUAL Algorithm, a left vertex in the decision phase is selected only if its buck per bang is lower than the currently largest price among the right vertices in the

reference set R' , and more importantly that the current largest price was derived from the buck per bang of a left vertex that arrived in the offline phase. With algorithm ON, a left vertex in the decision phase is selected as long as there is at least one unmatched right vertex with price larger than its buck per bang. Thus, if any left vertex is selected by VIRTUAL Algorithm then it is definitely selected by the ON algorithm. We illustrate the main difference between the ON and the VIRTUAL Algorithm via an example as follows.

Algorithm 3 Virtual Algorithm

```

1: %Offline Phase Input:  $L_t, (M_t, \gamma_t) = \text{THRESHOLD}(L_t \cup R)$ 
2:  $V = \{r : (\ell, r) \in M_t\}$ 
3:  $\text{price}(r) = b(e)$  for  $e = (\ell, r) \in M_t$ 
4: Order the elements of  $V$  in increasing  $\text{price}(r), r \in V$ ,
   the element with the largest price is  $r_{|V|}$ 
5: Initialize  $S = \Phi$ 
6: For every new left vertex  $\ell \in L \setminus L_t$  in the decision phase
7: if  $b(\ell) < \text{price}(r_{|V|})$  then
8:   if  $r_{|V|}$  was sampled in offline phase then
9:      $S = S \cup \{\ell\}$ 
10:  end if
11:  Update  $\text{price}(r_{|V|}) = \gamma(\ell)$ 
12:  Order the elements of  $V$  in increasing  $\text{price}(r), r \in V$ 
13: else Do nothing and keep  $\ell$  unmatched
14: end if

```

Example 1. Consider the input graph G , where in the offline phase two left vertices that are matched/selected by the THRESHOLD algorithm are $S = \{s_1, s_2\}$ with $\{b(s_1), b(s_2)\} = \{1/5, 1/6\}$. Let the left vertices ℓ_1, ℓ_2 (indexed in order of arrival) in the decision phase have $\{b(\ell_1), b(\ell_2)\} = \{1/5.1, 1/7\}$, respectively. Then with the ON algorithm, on arrival of ℓ_1 with $b(\ell_1) = 1/5.1$ it is compared with s_1 that has $b(s_1) = 1/5$ and since $b(\ell_1) < b(s_1)$, ℓ_1 is selected. Similarly, on arrival of ℓ_2 with $b(\ell_2) = 1/7$ it is compared with s_2 (that has not been compared before and matched) that has $b(s_2) = 1/6$, and ℓ_2 is also selected. With the VIRTUAL algorithm, the offline matched set $\{b(s_1), b(s_2)\} = \{1/5, 1/6\}$ remains the same as in ON. Moreover, in the decision phase, on arrival of ℓ_1 with $b(\ell) = 1/5.1$ it is compared with s_1 (with worst $b(\cdot)$ value among the two), and since $b(\ell_1) < b(s_1)$, ℓ_1 is selected. The main difference is in the next step, where the set $V = \{s_1, s_2\}$ is updated to include ℓ_1 and eject s_1 to get the reference set as $V = \{\ell_1, s_2\}$ with $\{b(\ell_1), b(s_2)\} = \{1/5.1, 1/6\}$. Next, when ℓ_2 arrives with $b(\ell) = 1/7$, even though it has better buck per bang than both $b(\ell_1)$ and $b(s_2)$, but since the maximum value of $b(\cdot)$ among ℓ_1 and s_2 , $1/5.1$ is seen in the decision phase and not in the offline phase; ℓ_2 is not selected.

From [3], with the VIRTUAL Algorithm, a new left vertex that appears at location i is selected if and only if at location i , the left vertex with the largest buck per bang in the virtual set V is sampled at or before time t . Since the permutations are uniformly random, the probability of this event is $\frac{t}{i-1}$. Hence

the probability of selecting $\ell \in L^*$ when it arrives at position $i \in [t+1, n]$ is

$$\begin{aligned} P(\ell \in L^* \text{ is selected}) &= \sum_{i=t+1}^n \frac{1}{n} \frac{t}{i-1} = \frac{t}{n} \sum_{i=t+1}^n \frac{1}{i-1} \\ &> \frac{t}{n} \int_t^n \frac{dx}{x} = \frac{t}{n} \ln\left(\frac{n}{t}\right), \end{aligned} \quad (1)$$

where the first equality follows since the probability of ℓ arriving at the i^{th} location is $\frac{1}{n}$ independent of i . Choosing $t = \frac{n}{e}$, maximizes the lower bound, and we get that $P(\ell \in L^* \text{ is selected}) = 1/e$.

Hence by linearity of expectation, we get that the expected value of the selected left vertices by ON algorithm is at least

$$\mathbb{E}\{v(\text{M}_{\text{ON}})\} \geq \sum_{\ell \in L^*} \frac{1}{e} v(\ell) = \frac{1}{e} v(\text{Moff}). \quad (2)$$

Now we enforce the condition that $w(\ell) < \text{cost}(r)$ for selecting a left vertex. We show in Lemma 5 that selecting left vertices only when $w(\ell) < \text{cost}(r)$ implies that ON algorithm satisfies the sum-weight constraint C . Recall that we have assumed that under the secretarial model of input, given $b(i) > b(j)$, $P(w(i) > w(j)) = \frac{1}{2}$. Since each left vertex ℓ selected by ON algorithm has $b(\ell) \leq b(j)$ for some left vertex j that is part of offline matching M_t . Thus, each left vertex that belongs to M_{ON} without enforcing $w(\ell) < \text{cost}(r)$, is selected with probability $1/2$ even when the constraint is enforced, and we get from (2), that

$$\mathbb{E}\{v(\text{M}_{\text{ON}})\} = \frac{1}{2e} v(\text{Moff}). \quad (3)$$

Finally, the result follows since $v(\text{Moff}) > \frac{v(\text{OPT})}{1+o(1)}$ from Corollary 1. ■

Lemma 5. *Algorithm ON satisfies the capacity constraint.*

Proof: Let $\gamma_t = \gamma_C(G_t)$ for simplicity. For each $r \in R'$ (right vertices matched in the offline phase), from Remark 1 we have that for THRESHOLD algorithm, $\sum_{r \in R'} \text{cost}(r) \leq C$. In the decision phase, any left vertex is accepted (is matched to $r \in R'$) if its weight is less than the cost of $r \in R'$, and once $r \in R'$ is matched it is not available thereafter (at most $|R'|$ left vertices are selected). Therefore, it directly follows that for the set of matched left vertices in the decision phase L_D , $\sum_{\ell \in L_D} w(\ell) \leq \sum_{r \in R'} \text{cost}(r)$. Since we know that $\sum_{r \in R'} \text{cost}(r) \leq C$, the claim follows. ■

Discussion: In this section, we proposed an online algorithm ON for the knapsack problem with competitive ratio $2e$, improving upon the currently best known bound of $10e$ [3], under an extra large market assumption (Assumption 1). Assumption 1 is reasonable for most networking applications and has been considered widely in auction literature [18]–[20]. Assumption 1 is also satisfied if the value of items is generated according to a stochastic process that is light-tailed, which is what is generally observed in practice. In the next section, we build upon the ON algorithm to propose a truthful algorithm for the online bipartite budgeted matching problem.

III. TRUTHFUL BUDGETED BIPARTITE MATCHING

Motivated by crowdsourcing and D2D communication applications, in this section, we consider an online matching

problem over a bipartite graph $G(L \cup R, E)$, where the right vertex set R is known ahead of time, while left vertices of L arrive sequentially in a random order. The incident edge utilities $v(e), e = (\ell, r), r \in R$ from a vertex $\ell \in L$ to set R are revealed only upon its arrival, as well as its bid $c(\ell)$, and the problem is to decide which unmatched vertex of R to match with ℓ , if at all, immediately and irrevocably. If vertex ℓ is matched, a payment p_ℓ is made to vertex ℓ that has to be at least as much as its reported bid $c(\ell)$. A total budget constraint of C is assumed for payments to be made to the matched left vertices. We assume that left vertices are strategic players, which could potentially manipulate the reporting of their true cost, and hence seek a truthful algorithm, i.e., no incoming vertex has incentive to misreport its bid. We continue to work under the secretarial model of input and the large market assumption (Assumption 1).

Remark 3. *As shown in [9], if bids of left vertices are used as payments, there is incentive for left vertices to misreport their bids, and consequently the mechanism is not truthful or incentive compatible. Thus, the payment strategy is non-trivial.*

Assumption 2. *In the secretarial (uniformly random) left vertex arrival model, we also assume that for two different edges e_1 and e_2 with distinct left vertices ℓ_1 and ℓ_2 arriving at locations $\pi(1)$ and $\pi(2)$, if $v(e_1) > v(e_2)$, then $P(c(\ell_1) < c(\ell_2)) = 1/2$.*

To solve the online truthful budgeted matching problem we propose the ON-TRUTH algorithm that is almost identical to the ON algorithm in terms of when a left vertex is selected. The first difference is in size t of the set of left vertices over which the offline algorithm THRESHOLD is run. With ON, $t = n/e$, while with ON-TRUTH, $t = \text{Binomial}(n, 1/2)$. The second difference is setting the reward for a right vertex that is part of the offline matching to be equal to the value of the matched edge, instead of the buck-per-bang as in ON. A new feature with ON-TRUTH is the payment rule for any selected left vertex, and the payment for left vertex ℓ of the selected edge e^* is $\gamma_C(G')v(e^*)$.

We first compute the expected utility of matching $\text{M}_{\text{ON-T}}$ produced by algorithm ON-TRUTH without enforcing the condition $c(\ell) \leq \text{cost}(r)$ for selecting a left vertex on Line 17, where the expectation is over the uniformly random left vertex arrival sequences.

Lemma 6. $\mathbb{E}\{v(\text{M}_{\text{ON-T}})\} \geq v(\text{OPT}(C))/12$, when condition $c(\ell) \leq \text{cost}(r)$ is not enforced for selecting a left vertex in ON-TRUTH.

To prove the result, we work with two intermediate algorithms SIMULATE and SAMPLEANDPERMUTE, that will help in lower bounding the utility of the matching $\text{M}_{\text{ON-T}}$ produced by ON-TRUTH, similar to [16]. The connection between SAMPLEANDPERMUTE and the proposed algorithm ON-TRUTH, is that the output matching M_{3p} of SAMPLEANDPERMUTE and $\text{M}_{\text{ON-T}}$ produced by ON-TRUTH are almost identical, except for the difference in defining the set L' (set of left vertices used to generate the threshold γ'), without enforcing condition $c(\ell) \leq \text{cost}(r)$ in ON-TRUTH. But with both these definitions,

Algorithm 4 ON-TRUTH Algorithm

```

1: Input:  $L$  set of left vertices/users that arrive sequentially
   with permutation  $\pi$ ,  $R$  set of right vertices, Payment
   Budget  $C$ 
2: %Offline Phase
3:  $p = \frac{1}{2}$ ,  $k \leftarrow \text{Binomial}(|L|, p)$ 
4: Let  $L'$  be the first  $k$  vertices of  $L$ 
5: Run THRESHOLD on  $G' = (L' \cup R, E')$  to obtain  $\gamma' \triangleq \gamma_C(G')$  and matching  $M_1$ 
6: for each right vertex  $r : e = (\ell, r) \in M_1$  do
7:   Set  $\text{reward}(r) := v(e)$  and  $\text{cost}(r) := c(\ell)$ 
8: end for
9: for each right vertex  $r : (*, r) \notin M_1$  do
10:   Set  $\text{reward}(r) := 0$  and  $\text{cost}(r) := 0$ 
11: end for
12: %Decision Phase
13:  $M_{\text{ON-T}} = \emptyset$ 
14: for every new left vertex  $\ell \in L \setminus L'$  do,
15:   %Pruning: Delete all edges  $e = (\ell, r), r \in R$  s.t.
      $b(e) > \gamma'$ 
16:   Let  $e^* = (\ell, r)$  be the edge with the largest value such
     that  $v(e^*) \geq \text{reward}(r)$  AND  $c(\ell) \leq \text{cost}(r)$ 
17:   if  $M_{\text{ON-T}} \cup \{e^*\}$  is a matching then
18:      $M_{\text{ON-T}} = M_{\text{ON-T}} \cup \{e^*\}$ 
19:     Pay  $p_\ell = \gamma' v(e^*)$  to vertex  $\ell$ 
20:   else
21:     Let  $\ell$  be permanently unmatched
22:   end if
23: end for

```

Algorithm 5 SIMULATE Algorithm

```

1: Input: Graph  $G$  and threshold  $(\gamma)$ 
2: Output: Matching  $M_{1s}, M_{2s}$ 
3: Remove edges of  $G$  with  $b(e) > \gamma$  to get  $G(\gamma)$ 
4: Sort edges of  $G(\gamma)$  in decreasing order of their value
5:  $M_{1s} = \Phi, M_{2s} = \Phi$ 
6: Mark each left vertex  $\ell \in G(\gamma)$  as unassigned
7: For each edge  $e = (\ell, r)$  in sorted order
8: if  $\ell$  is unassigned AND  $M_{1s} \cup e$  is a matching then
9:   Mark  $\ell$  as assigned
10:   Flip a coin with probability  $p$  of heads
11:   If heads,  $M_{1s} \leftarrow M_{1s} \cup e$ 
12:   else  $M_{2s} \leftarrow M_{2s} \cup e$ 
13: end if

```

a left vertex is selected to be part of L' with probability $1/2$ independently. Consequently, the utilities of matchings M_{3p} and $M_{\text{ON-T}}$ are identical in expectation. So to lower bound the utility of $M_{\text{ON-T}}$, we find a lower bound on the utility of M_{3p} of SAMPLEANDPERMUTE, and focus entirely on SAMPLEANDPERMUTE algorithm as follows.

SIMULATE is an offline matching algorithm, where each edge in descending order of its value is either assigned to matching M_{1s} or pseudo matching M_{2s} ² depending on the coin toss for that edge. Important to notice is that for SIMULATE,

² M_{2s} is not a matching since in M_{2s} multiple edges can be incident on any right vertex.

Algorithm 6 SampleAndPermute Algorithm

```

1: Input: Graph  $G = (L \cup R, E)$ 
2: Output: Matching  $M_{2p}, M_{3p}$ 
3: %Offline Phase
4:  $L' = \Phi$ 
5: for each  $\ell \in L$  do
6:   With probability  $\frac{1}{2}$ ,  $L' \leftarrow L' \cup \ell$ 
7: end for
8:  $(M_{1p}, \gamma') \rightarrow \text{THRESHOLD}(G(L' \cup R, E(L')))$ 
9: for each  $r \in R$  do
10:   Set  $\text{reward}(r) = v(e)$  if  $e = (\ell, r) \in M_{1p}$ 
11:   Set  $\text{reward}(r) = 0$  if  $e = (*, r) \notin M_{1p}$ 
12: end for
13: %Decision Phase
14:  $M_{2p} = \Phi, M_{3p} = \Phi$ 
15: for each  $\ell \in L \setminus L'$  and  $b(e) \leq \gamma'$  do in random order
16:   Let  $e = (\ell, r)$  be the edge with largest value such that
      $v(e) \geq \text{reward}(r)$ 
17:   Add  $e$  to  $M_{2p}$ .
18:   If  $M_{3p} \cup e$  is a matching  $M_{3p} \leftarrow M_{3p} \cup e$ 
19: end for

```

once a coin is tossed for an edge making the left vertex assigned, no other coin is tossed for any edge that shares a common left vertex with it. So it is essentially identical to tossing a coin once for each left vertex instead of each individual edge as done in algorithm SAMPLEANDPERMUTE. Thus, whenever coin tosses are identical for SAMPLEANDPERMUTE and SIMULATE, and the γ' computed by THRESHOLD algorithm invoked inside SAMPLEANDPERMUTE is identical to the input γ to SIMULATE, it is easy to see that the matching $M_{1s} = M_{1p}$ and pseudo matching $M_{2s} = M_{2p}$ produced by SIMULATE and SAMPLEANDPERMUTE [16].

Lemma 7 (Lemma 2.3 [16]). *For SIMULATE algorithm, if the input threshold γ and the coin tosses for choosing an edge (to be part of M_{1s} or M_{2s}) are independent, then $\mathbb{E}\{v(M_{2s})\} = \mathbb{E}\{v(M_{1s})\}$.*

Remark 4. *Lemma 6 would be directly provable following the techniques of [16], if Lemma 7 could be applied on the matching M_{1s} and M_{2s} , for the case when $M_{1s} = M_{1p}$ and pseudo matchings $M_{2s} = M_{2p}$. Problem is that $M_{1s} = M_{1p}$ and pseudo matchings $M_{2s} = M_{2p}$ only when the respective coin tosses in SIMULATE and SAMPLEANDPERMUTE, and the γ (input to SIMULATE) and γ' (computed by THRESHOLD algorithm invoked inside SAMPLEANDPERMUTE) are identical. Since γ' is dependent on coin tosses of SAMPLEANDPERMUTE, so if γ' in input to SIMULATE and the coin tosses are identical to as in SAMPLEANDPERMUTE, they are dependent on each other, and Lemma 7 is not applicable.*

So the proof of Lemma 6 is more involved as presented next.

Proof: Consider the full graph $G = (L \cup R, E)$ (offline), and let γ_f be the output threshold when THRESHOLD is run over the full graph G .

Toss $2|L|$ coins independently with heads probability $1/2$,

and record their outcomes in two vectors $\mathbf{t}_1 = [t_{11} \dots t_{1|L|}]$ and $\mathbf{t}_2 = [t_{21} \dots t_{2|L|}]$, where $t_{ij} = 1$ if the $(i, j)^{th}$ coin toss is heads, and 0 otherwise. We will associate \mathbf{t}_1 with coin tosses for the $|L|$ left vertices while running SIMULATE with full graph G and threshold γ_f .

All left vertices for which $t_{2j} = 1$ (set L' as defined in SAMPLEANDPERMUTE), will be part of the offline phase and the remaining left vertices with $t_{2j} = 0$ will be part of decision/online phase in algorithm SAMPLEANDPERMUTE. Let γ'_{t_2} be the output threshold from THRESHOLD when executed inside the algorithm in SAMPLEANDPERMUTE with coin toss vector \mathbf{t}_2 , which is identical to running algorithm THRESHOLD on graph $(G(L' \cup R, E'))$.

Remark 5. Note that for fixed coin tosses \mathbf{t}_2 that determines γ'_{t_2} completely, the matchings M_{1p}, M_{2p} produced by SAMPLEANDPERMUTE are identical to the matchings M_{1s}, M_{2s} produced by SIMULATE with input $G(L \cup R, E)$ and threshold γ'_{t_2} , and coin tosses \mathbf{t}_2 , respectively. Hence

$$\mathbb{E}\{v(M_{2p}(G(\gamma_t)))\} = \mathbb{E}\{v(M_{2s}(G(\gamma_t)))\}^3 \quad (4)$$

For fixed realizations of coin tosses \mathbf{t}_1 and \mathbf{t}_2 , now we compare the pseudo matchings M_{2s} produced by algorithm SIMULATE with input graph $G = (L \cup R, E)$, threshold γ_f with coin tosses \mathbf{t}_1 , and input graph $G(L' \cup R, E')$ and threshold γ_{t_2} with coin tosses \mathbf{t}_2 , respectively.

Let the set of left vertices that have at least one edge in $G(\gamma_f)$ and in $G(\gamma_{t_2})$ be L_1 and L_2 , respectively. Recall that $G(\gamma)$ contains all edges $e \in G$ that have $b(e) \leq \gamma$. From Lemma 2, we know that any choice of \mathbf{t}_2 , $\gamma_{t_2} \geq \gamma_f$. Hence $L_1 \subseteq L_2$. Consider the case when the realization of coin tosses \mathbf{t}_1 and \mathbf{t}_2 restricted to set L_1 of left vertices be the same. Then independent of the coin tosses for vertices $L_2 \setminus L_1$, we have that for algorithm SIMULATE

$$v(M_{2s}(G(\gamma_{t_2}))) \geq v(M_{2s}(G(\gamma_f))), \quad (5)$$

since $\gamma_t \geq \gamma_f$ and each edge present in $G(\gamma_f)$ is also present in $G(\gamma_{t_2})$ and M_{2s} is a pseudo matching and accepts multiple edges incident on any right vertex. For pseudo matching we let $v(M_{2s}(G(\gamma))) = \sum_{e \in M_{2s}(G(\gamma))} v(e)$. Thus, taking expectation of (5), we have that

$$\mathbb{E}\{v(M_{2s}(G(\gamma_{t_2})))\} \geq \mathbb{E}\{v(M_{2s}(G(\gamma_f)))\}. \quad (6)$$

Since γ_f (obtained by running THRESHOLD on full graph G) does not depend on any coin tosses \mathbf{t}_1 or \mathbf{t}_2 , we have from Lemma 7,

$$\mathbb{E}\{v(M_{2s}(G(\gamma_f)))\} = \mathbb{E}\{v(M_{1s}(G(\gamma_f)))\}, \quad (7)$$

where the expectation is over the coin tosses \mathbf{t}_1 .

Next, we lower bound the $\mathbb{E}\{v(M_{1s}(G(\gamma_f)))\}$. Consider graph G and threshold γ_f as an input to the SIMULATE algorithm. For a fixed realization of \mathbf{t}_1 , let $\text{OPT}_{1/2}(\mathbf{t}_1)$ be the optimal matching considering only left vertices j for which coin tosses $t_{1j} = 1$. Taking the expectation with respect to \mathbf{t}_1 , we have that

$$\mathbb{E}\{v(\text{OPT}_{1/2})\} = \frac{\mathbb{E}\{v(\text{OPT})\}}{2}. \quad (8)$$

³ $M(G(\gamma))$ is the matching obtained with graph G and threshold γ .

Moreover, as pointed out earlier in Remark 1, the matching produced by THRESHOLD with output threshold $\hat{\gamma}$ is equivalent to finding a greedy matching with graph $G(\hat{\gamma})$. The same is true for matching M_{1s} produced by SIMULATE. Thus, considering SIMULATE algorithm with input G and threshold γ , and all left vertices with $t_{1j} = 1$ with \mathbf{t}_1 , from Lemma 1, $v(M_{1s}(G(\gamma))) \geq \frac{v(\text{OPT}_{1/2}(\mathbf{t}_1))}{3}$. Thus, taking the expectation with respect to \mathbf{t}_1 , from (8), we get

$$\mathbb{E}\{v(M_{1s}(G(\gamma)))\} = \frac{v(\text{OPT}(C))}{6}. \quad (9)$$

From Lemma 2.5 [16], we have that for the SAMPLEANDPERMUTE algorithm,

$$\mathbb{E}\{v(M_{3p}(G(\gamma_t)))\} \geq \frac{\mathbb{E}\{v(M_{2p}(G(\gamma_t)))\}}{2}, \quad (10)$$

since the pruning step to obtain M_{3p} from M_{2p} in SAMPLEANDPERMUTE algorithm only depends on the relative order in which vertices (with coin tosses $t_{2j} = 0$) arrive in the decision phase, and not on coin tosses themselves.

Combining (6), (7), (9), (10), and (4) we get that

$$\mathbb{E}\{v(M_3(\gamma_t))\} \geq \frac{v(\text{OPT}(C))}{12}.$$

Since the expected utility of matching $M_3(\gamma_{t_2})$ of SAMPLEANDPERMUTE is same as the expected utility of matching $M_{\text{ON-T}}$ (the output of ON-TRUTH), we have the result. ■

The following theorem is the second main result of the paper.

Theorem 2. Algorithm ON-TRUTH is 24-competitive, satisfies the payment budget constraint, payment is always larger than the bid for each selected left vertex, i.e., $p_\ell \geq c(\ell)$, and is truthful.

Proof: Disregarding the condition $c(\ell) < \text{cost}(r)$, an edge e incident on a left vertex ℓ is chosen by algorithm ON-TRUTH if its value $v(e)$ is larger than the reward (value of an edge of a right vertex that is part of the offline matching). Hence from Assumption 2, if a left vertex is accepted without the condition $c(\ell) < \text{cost}(r)$, then it is still accepted with probability $\frac{1}{2}$ while enforcing the condition $c(\ell) < \text{cost}(r)$. Combining this fact with Lemma 6, we get the 24-competitiveness of ON-TRUTH algorithm.

The claim that $p_\ell \geq c(\ell)$ for each selected left vertex ℓ , follows from the fact that each left vertex is considered in the decision phase only if its buck per bang $b(e) = \frac{c(\ell)}{v(\ell)} < \gamma$. Since $p_\ell = \gamma v(\ell)$, clearly, $p_\ell \geq c(\ell)$. The budget feasibility and incentive compatibility are shown in Lemma 8 and 9, respectively. ■

Lemma 8. Algorithm ON-TRUTH satisfies the payment budget constraint.

Proof: Similar to Lemma 5, by enforcing the condition that any left vertex is selected only if $c(\ell) \leq \text{cost}(r)$ for some unmatched right vertex that is part of M_1 , and $\sum_{r: e=(\ell, r) \in M_1} \text{cost}(r) \leq C$ for the THRESHOLD algorithm by Remark 1. ■

Next, we show the most important property of ON-TRUTH, its truthfulness. Towards that end, we will use the Myerson's Theorem [21].

Theorem 3. [21] A reverse auction is truthful if and only if:

- The selection rule is monotone. If a user ℓ wins the auction by bidding $c(\ell)$, it would also win the auction by bidding an amount $c(\ell)'$, where $c(\ell)' < c(\ell)$.
- Each winner is paid a critical amount. If a winning user submits a bid greater than this critical value, it will not get selected.

Lemma 9. ON-TRUTH is a truthful online algorithm.

Proof: We show that the two conditions of Theorem 3 are satisfied for the ON-TRUTH algorithm, similar to [17]. In the decision phase, if any left vertex reduces its bid, then clearly its buck per bang $b(e)$ decreases, and hence it is still accepted if it was accepted before. Thus, monotone condition is satisfied.

The criticality of payment is shown as follows. Note that the payment made by ON-TRUTH to a selected left vertex ℓ is $p_\ell = \gamma' v(e)$, $e = (\ell, r) \in M_{\text{ON-T}}$, where the right vertex index r is such that utility $v(e)$, $e = (\ell, r)$ is largest among the unmatched right vertices at the time of arrival of vertex ℓ that have an edge to left vertex ℓ , and $v(e) > \text{reward}(r)$.

Now, if suppose the bid $c(\ell)$ of left vertex ℓ is more than $p_\ell = \gamma' v(e)$, then its bang per buck $c(\ell)/v(e) > \gamma'$. Moreover, since $v(e) > v(e')$ for all edges e' incident on unmatched right vertices from ℓ at the arrival of left vertex ℓ , we have that $c(\ell)/v(e') > \gamma'$. Thus, all edges out of left vertex ℓ incident on currently unmatched right vertices are removed in the pruning stage of the decision phase, and hence vertex ℓ cannot be selected. ■

Discussion: In this section, we proposed a 24-competitive online algorithm for the budgeted bipartite matching problem that is also truthful, improving upon the best known bound of 24β -competitive [17], where β is the ratio of the maximum and minimum utility of any edge. The proposed algorithm is a significant/fundamental improvement over prior work, since it eliminates any dependence on the system/input parameters, making it scalable and suitable for large networks.

IV. SIMULATION

We consider the uplink of a single cell of cellular communication for the D2D application, where 150 cellulars users are present with one basestation. Out of 150 nodes, the helper set is of size $n = 50$, while the rest 100 nodes (set R) are seeking help. The payment budget constraint is 100. All users are assumed to be uniformly located in the coverage area, and the utility between any helper and a help seeking node is drawn uniformly from $[0, 20]$, and the bid for each helper is drawn uniformly from $[0, 5]$. Let δ be the fraction of nodes any one helper can help, and we assume that for fixed δ , the nodes that any helper can help are uniformly distributed among the 100 nodes. In Fig. IV, we plot the competitive ratio of the proposed algorithm ON-TRUTH as a function of δ . We see that the competitive ratio of ON-TRUTH algorithm is far better than the derived guarantee (24-competitive). An important observation from Fig. IV is that as δ increases, the competitive ratio increases significantly, since with larger δ , the quality of the offline matching and the number of right

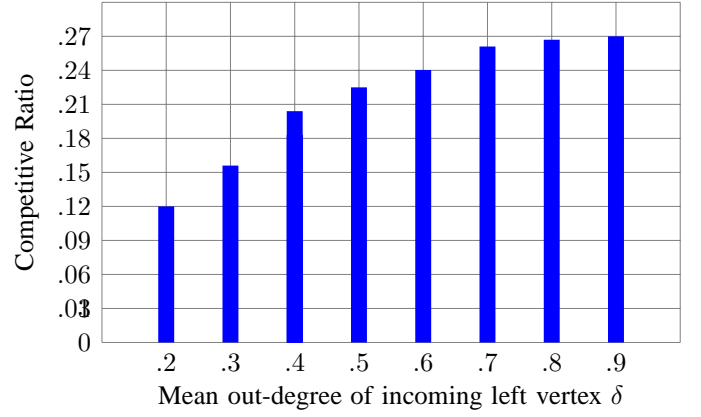


Fig. 1. Competitive ratio of the ON-TRUTH algorithm as a function of δ .

vertices matched in the offline phase increases, allowing the ON-TRUTH algorithm to match larger number of left vertices, and extract larger utility.

V. CONCLUSIONS

In this paper, we have made significant progress in finding better online algorithms for bipartite matching under the capacity constraint on the 'size' of selected left vertices. Under the large market assumption, that is reasonable in practice, we are able to improve the best known competitive ratio for the knapsack problem from $10e$ to $2e$, and from non-constant to constant for the truthful matching problem.

APPENDIX A PROOF OF LEMMA 1

Proof: Decompose the optimal fractional matching solution $\text{OPT} = \{\text{OPT}^+ \cup \text{OPT}^-\}$, where OPT^+ contains edges of OPT that have $b(e) > \gamma_C$, and OPT^- contains edges of OPT that have $b(e) \leq \gamma_C$. Similarly, let $\text{OPT}(\gamma_C)$ be the optimal fractional matching on subgraph $G(\gamma_C) \subseteq G$, where γ_C is the output threshold from the THRESHOLD algorithm with graph G . By definition of optimal matching, $v(\text{OPT}^-) \leq v(\text{OPT}(\gamma_C))$. Moreover, for M , the output matching from THRESHOLD algorithm with graph G , we have $v(M) \geq \frac{v(\text{OPT}(\gamma_C))}{2}$, since M is a greedy matching on $G(\gamma)$ (subgraph with all edges having $b(e) \leq \gamma_C$). Therefore, $v(M) \geq \frac{v(\text{OPT}^-)}{2}$.

All edges $e = (\ell, r) \in \text{OPT}^+$, have $b(e) = \frac{c(\ell)}{v(e)} > \gamma_C$. Thus, $v(\text{OPT}^+) = \sum_{e=(\ell,r) \in \text{OPT}^+} x(\ell)v(e) < \frac{\sum_{e=(\ell,r) \in \text{OPT}^+} x(\ell)c(\ell)}{\gamma_C}$, where $x(\ell)$ are fractional weights in the optimal solution. Moreover, the total budget constraint of C ($\sum_{e=(\ell,r) \in \text{OPT}} x(\ell)c(\ell) \leq C$) implies that $v(\text{OPT}^+) < \frac{C}{\gamma_C}$. Assuming that the budget constraint is tight with the THRESHOLD algorithm ($\sum_{e \in M} \gamma_C v(e) = C$), $v(M) = \frac{C}{\gamma_C}$. Therefore, $v(\text{OPT}^+) < v(M)$. Combining this with $v(M) \geq \frac{v(\text{OPT}^-)}{2}$, we have $v(M) \leq 3v(\text{OPT})$ as required.

If the capacity constraint is not tight with the THRESHOLD algorithm, then under Assumption 1 ($v_{\max}/v(\text{OPT}) = o(1)$), by the definition of THRESHOLD algorithm that finds the largest feasible γ , the leftover capacity $C - \sum_{e \in M} \gamma_C v(e)$

is no more than $\gamma_C v_{max}$, and similar argument gives us that $v(\text{OPT}) \leq (3 + o(1))v(M)$. ■

APPENDIX B PROOF OF COROLLARY 1

Proof: Compared to the proof of Lemma 1, the only difference is in contribution from edges with $b(e) \leq \gamma_C$. Since all edge weights incident on any left vertex are identical and the number of right vertices is equal to left vertices, greedy matching M is actually optimal for edges with $b(e) \leq \gamma_C$. Hence $v(M) = v(\text{OPT}(\gamma_C))$. Noting that $v(\text{OPT}(\gamma_C)) \geq v(\text{OPT}^-)$, we get $v(M) \geq v(\text{OPT}^-)$, and from which the result follows, since $v(M) \geq v(\text{OPT}^+)$ (proof of Lemma 1). ■

APPENDIX C PROOF OF LEMMA 2

Proof: When a left vertex is removed (by deleting all edges incident to it as considered), the proof of claim 1 follows by standard procedure by considering each right vertex, for which the value of the matched edge in $\text{GREEDY}(G)$ is at least as much as in $\text{GREEDY}(F)$.

For the second and third claim, note that an edge e incident on left vertex ℓ is removed in $G(\gamma)$ compared to G , if $b(e) > \gamma$ or equivalently if $v(e) < \frac{c(\ell)}{\gamma}$. Recall that the cost of any edge only depends on the index of its left vertex. Hence, if edge $e = (\ell, r)$ is removed from G to obtain $G(\gamma)$, then all the edges e' incident on ℓ with utility $v(e') < v(e)$ are also removed. So essentially, edges are removed monotonically from G to produce $G(\gamma)$. So the proofs for the second and third claim follow similarly to the first. ■

APPENDIX D PROOF OF LEMMA 3

Proof: From the definition of Algorithm THRESHOLD it's clear that if any $\gamma \in \mathcal{A}(G)$, then $\gamma_C \geq \gamma$. Hence the key step is to show that if any $\gamma \notin \mathcal{A}(G)$, then $\gamma_C < \gamma$ which follows from the second claim of Lemma 2, that $v(\text{GREEDY}(G(\gamma_1))) \geq v(\text{GREEDY}(G(\gamma_2)))$ for $\gamma_1 \geq \gamma_2$. Therefore, if for any $\gamma \notin \mathcal{A}(G)$, then for any $\gamma' > \gamma$, $\gamma' \notin \mathcal{A}(G)$. Hence we can use bisection to find the maximum. ■

APPENDIX E PROOF OF LEMMA 4

Proof: From Lemma 2,

$$v(M(F(\gamma))) \leq v(\text{GREEDY}(G(\gamma))). \quad (11)$$

Let the threshold and the matching obtained by running THRESHOLD on G with budget C be $\gamma_C(G) = \gamma$, and $M(G)$, respectively, where $\gamma \leq \frac{C}{v(\text{GREEDY}(G(\gamma)))}$. Now we consider $F(\gamma)$ as the input graph to the THRESHOLD with same budget constraint C . Since $\gamma \leq \frac{C}{v(\text{GREEDY}(G(\gamma)))}$, from (11), clearly, $\gamma \leq \frac{C}{v(M(F(\gamma)))}$, and $\sum_{e \in M(F(\gamma))} \gamma v(e) \leq C$. Therefore, $\gamma \in \mathcal{A}(F)$, which by definition of $\gamma_C(F)$ implies $\gamma_C(F) \geq \gamma$. ■

REFERENCES

- [1] A. Marchetti-Spaccamela and C. Vercellis, "Stochastic on-line knapsack problems," *Mathematical Programming*, vol. 68, no. 1-3, pp. 73–104, 1995.
- [2] X. Han, Y. Kawase, and K. Makino, "Randomized algorithms for online knapsack problems," *Theoretical Computer Science*, vol. 562, pp. 395–405, 2015.
- [3] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, "A knapsack secretary problem with applications," in *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*. Springer, 2007, pp. 16–28.
- [4] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp. 71–83, 2014.
- [5] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 433–441.
- [6] Z. Zheng, M. Li, X. Xiao, and J. Wang, "Coordinated resource provisioning and maintenance scheduling in cloud data centers," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 345–349.
- [7] M. Cello, G. Gnecco, M. Marchese, and M. Sanguineti, "A generalized stochastic knapsack problem with application in call admission control," in *10th Cologne-Twente Workshop on Graphs and Combinatorial Optimization CTW 2011*, p. 105.
- [8] Y. Zhang and C. Leung, "Resource allocation in an OFDM-based cognitive radio system," *IEEE Transactions on Communications*, vol. 57, no. 7, pp. 1928–1931, 2009.
- [9] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 173–184.
- [10] A. Subramanian, G. S. Kanth, S. Moharir, and R. Vaze, "Online incentive mechanism design for smartphone crowd-sourcing," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2015 13th International Symposium on*. IEEE, 2015, pp. 403–410.
- [11] N. Anari, G. Goel, and A. Nikzad, "Mechanism design for crowdsourcing: An optimal 1-1/e competitive budget-feasible mechanism for large markets," in *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE, 2014, pp. 266–275.
- [12] G. Goel, A. Nikzad, and A. Singla, "Matching workers expertise with tasks: Incentives in heterogeneous crowdsourcing markets," in *NIPS Workshop on Crowdsourcing*, 2013.
- [13] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1801–1819, 2014.
- [14] W. Saad, Z. Han, M. Debbah, and A. Hjørungnes, "A distributed merge and split algorithm for fair cooperation in wireless networks," in *ICC Workshops-2008 IEEE International Conference on Communications Workshops*. IEEE, 2008, pp. 311–315.
- [15] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: fundamentals and applications," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 52–59, 2015.
- [16] N. Korula and M. Pál, "Algorithms for secretary problems on graphs and hypergraphs," in *Automata, Languages and Programming*. Springer, 2009, pp. 508–520.
- [17] R. Vaze and M. Coupechoux, "Online budgeted truthful matching," in *NetEcon*, June 2016.
- [18] K. Iyer, R. Johari, and M. Sundararajan, "Mean field equilibria of dynamic auctions with learning," *Management Science*, vol. 60, no. 12, pp. 2949–2970, 2014.
- [19] R. Gomes and V. Mirrokni, "Optimal revenue-sharing double auctions with applications to ad exchanges," in *Proceedings of the 23rd international conference on World Wide Web WWW*. ACM, 2014, pp. 19–28.
- [20] M. Mihailescu and Y. M. Teo, "On economic and computational-efficient resource pricing in large distributed systems," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2010, pp. 838–843.
- [21] R. B. Myerson, "Optimal auction design," *Mathematics of operations research*, vol. 6, no. 1, pp. 58–73, 1981.
- [22] Y. Singer and M. Mittal, "Pricing mechanisms for crowdsourcing markets," in *Proceedings of the 22nd International Conference on World Wide Web*, 2013, pp. 1157–1166.

- [23] D. Zhao, X. Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 1213–1221.